of the number of ground facts in the knowledge base. It is easy to show that the data complexity of forward chaining is polynomial, not exponential.

- We can consider subclasses of rules for which matching is efficient. Essentially every Datalog clause can be viewed as defining a CSP, so matching will be tractable just when the corresponding CSP is tractable. Chapter 6 describes several tractable families of CSPs. For example, if the constraint graph (the graph whose nodes are variables and whose links are constraints) forms a tree, then the CSP can be solved in linear time. Exactly the same result holds for rule matching. For instance, if we remove South Australia from the map in Figure 9.5, the resulting clause is

  $$Diff(wa,nt) \land Diff(nt,q) \land Diff(q,nsw) \land Diff(nsw,v) \Rightarrow Colorable()$$

  which corresponds to the reduced CSP shown in Figure 6.12 on page 201. Algorithms for solving tree-structured CSPs can be applied directly to the problem of rule matching.
- We can try to eliminate redundant rule-matching attempts in the forward-chaining algorithm, as described next.

Incremental forward chaining

When we showed how forward chaining works on the crime example, we cheated. In particular, we omitted some of the rule matching done by the algorithm shown in Figure 9.3. For example, on the second iteration, the rule

$$Missile(x) \Rightarrow Weapon(x)$$

matches against $Missile(M_1)$ (again), and of course the conclusion $Weapon(M_1)$ is already known so nothing happens. Such redundant rule matching can be avoided if we make the following observation: *Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration $t-1$.* This is true because any inference that does not require a new fact from iteration $t-1$ could have been done at iteration $t-1$ already.

This observation leads naturally to an incremental forward-chaining algorithm where, at iteration t, we check a rule only if its premise includes a conjunct p_i that unifies with a fact p'_i newly inferred at iteration $t-1$. The rule-matching step then fixes p_i to match with p'_i, but allows the other conjuncts of the rule to match with facts from any previous iteration. This algorithm generates exactly the same facts at each iteration as the algorithm in Figure 9.3, but is much more efficient.

With suitable indexing, it is easy to identify all the rules that can be triggered by any given fact, and many real systems operate in an "update" mode wherein forward chaining occurs in response to every TELL. Inferences cascade through the set of rules until the fixed point is reached, and then the process begins again for the next new fact.

Typically, only a small fraction of the rules in the knowledge base are actually triggered by the addition of a given fact. This means that a great deal of redundant work is done in repeatedly constructing partial matches that have some unsatisfied premises. Our crime example is rather too small to show this effectively, but notice that a partial match is constructed on the first iteration between the rule

$$American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$$

and the fact $American(West)$. This partial match is then discarded and rebuilt on the second iteration (when the rule succeeds). It would be better to retain and gradually complete the partial matches as new facts arrive, rather than discarding them.